

EAST SEARCH 01/19/06 TM *James Menden*

	Type	Hits	Search Text	DBs
1	BRS	393	712/215.ccls.	USPAT
2	BRS	544	712/228,229.ccls.	USPAT
3	BRS	178	712/43.ccls.	USPAT
4	BRS	1	paraday.in.	USPAT
5	BRS	8	paraday	USPAT
6	BRS	2721	SMT	USPAT
7	BRS	76	simultaneous adj multithread\$3	USPAT
8	BRS	55	SMT and (resource near2 shar\$3)	USPAT
9	BRS	118	SMT and (resource near2 shar\$3)	US - PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB
10	BRS	69	SMT and (resource near2 shar\$3) and thread\$3	US - PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB
11	BRS	4	("5574928" "5812811" "5881307" "6154831").PN.	US - PGPUB; USPAT; USOCR
12	BRS	1281	(thread\$4) and (switch\$3 with ((long adj latency) or stall\$3 or (miss)))	US - PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB
13	BRS	246	S12 and ((multithread\$3) or (multi adj thread\$3))	US - PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB
14	BRS	44	S12 and ((multithread\$3) or (multi adj thread\$3)) and SMT	US - PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB

	Type	Hits	Search Text	DBs
15	BRS	146	eggers.in. and thread\$3	US - PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB
16	BRS	90	eggers.in. and thread\$3 and switch\$3	US - PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB
17	BRS	4	eggers.in. and thread\$3 and switch\$3 and (latency or (cache adj miss))	US - PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB
18	BRS	281	simultaneous adj multithreading	US - PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB
19	BRS	322	simultaneous adj multithread\$3	US - PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB
20	BRS	322	simultaneous adj multithread\$3	US - PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB

	Type	Hits	Search Text	DBs
21	BRS	53	(simultaneous adj multithread\$3) and (switch\$3 with ((cache near2 miss) or (long near2 latency)))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB

*NORTH SEARCH 01/19/06 TM
8 PAGES*

Information Security · Networks · Processor
Architecture
Memory Access and Control
General Resources · NORTH Help

[Send](#)
[Comment](#)

Admin Log-in · Search:

GO

Dmitri Mendeleev

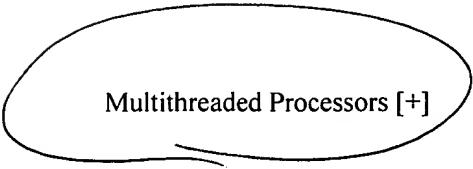
Processor Architecture

This unit covers advanced topics within processor architecture. It looks at structures of both uniprocessor systems and multiprocessor systems.

Overview: Processor Design · Advanced Topics · Survey of High Performance Architectures · CPU Organization Overview · Advanced Topics in Microarchitecture

Prerequisites: Computer Architecture

Topics	References	Tutorials	Patents
Pipelining (Stall, Bubble, Multiple, Flush) [+]		<ul style="list-style-type: none"> • Tutorial: Pipelining • Pipelining • Instruction Pipelining • Advanced Issues in Pipelining 	
Superscalar Architecture [+]	<ul style="list-style-type: none"> • Asynchronous Superscalar Architecture • Superscalar Processor Architecture 	<ul style="list-style-type: none"> • The Superscalar Hardware Architecture of the MC68060, Joseph Circello, Computer Museum History Center, Video (25 minutes), Hot Chips VI - 1994 [8/15/1994 1 PM (Pacific)] 	
VLIW [+]		<ul style="list-style-type: none"> • VLIW/Superscalar Processors 	
Multiprocessors [+]		<ul style="list-style-type: none"> • Multiprocessor Systems • Parallel Processors • Scalable Multiprocessors and the DASH Approach John Hennessy Video (52 minutes) Computer Museum History Center 4/10/1992 1 PM (Pacific) • Cache-Coherent Multiprocessors: an Easy Approach to High-Performance Computing Forest Baskett 	



Multithreaded Processors [+]

- Multithreading documents list
- M. J. Flynn and A. Podvin, *Shared Resource Multiprocessing*, IEEE Computer, pp. 20-28, March 1972.
- The MAJC Processor Architecture, Marc Tremblay, Video (60 minutes), Stanford University, [3/29/2000 4:15 AM (Pacific)]
- Multithreaded processors lecture material
- Radical Chip Multithreading lecture material
- Trace Processors Reference list
- Trace Processors
- Rotenberg, Eric, *Trace Processors: Exploiting Hierarchy and Speculation*, Doctorial Thesis, University of Wisconsin-Madison, 1999.
- FPGA Reference list
- DRHW WWW Library
- Adaptive hardware becomes a reality using electrically reconfigurable arrays (ERAs)
- The Nano Processor: a low resource reconfigurable processor
- The implementation of hardware subroutines on field programmable gate arrays
- Memory Hierarchy in Cache Based Systems
- Viddeo (40 minutes)
Computer Museum History Center
10/18/1990 1 PM (Pacific)
- 6697935
- 6694347
- 6567839
- 6535905
- 6507862
- 6341347
- 6105127
- 6076157
- 6049867
- 5933627
- 5913049
- 5742782
- 5430851
- 5361337
- 6240509
- 6216206
- 6185675
- 6182210
- 6170038
- 6073213
- 6018786
- 5381533
- 6721884
- 6182206
- 6023755
- 5968161
- 5956518
- 5684980
- Reconfigurable Computing
- Memory Hierarchy Overview

Trace Processing [+]

Reconfigurable Processors, FPGA [+]

Memory Hierarchy* [+]

- CPUs combined with bulk memory reference list
 - Memory accessing (e.g. prefetch) and RAMBUS reference list
 - Vector Processors Overview
 - Vector pipelining, chaining, and speed on the IBM 3090 and Cray X-MP
 - SIMD Instructions
 - SIMD Architectures
 - MIMD Architectures
- Vector Processors [+]
- SIMD [+]
- MIMD [+]
- Vector Processing

Advanced Units: Instruction Sets · Memory Data Flow Techniques · Register Data Flow Techniques

* Denotes a topic that is useful but not necessary.

NORTH: North Online Relational Training Hierarchy
Developed by: Nick Galotti, Justin Marrese, Maulin Patel, Mike Pyzocha as part of a project for WPI

Slides from technical training seminars

Section Index:

[Year 2004 Technical Training](#)

[References from Trace processors lecture](#)

[References from multithreaded processor lecture](#)

See Also:

[Additional threaded processor documents](#)

[Radical Chip Multithreading lecture](#)

Title Index (19 references):

- [A Survey of Processors with Explicit Multithreading](#)
 - [A multithreaded PowerPC processor for commercial servers](#)
 - [Audio recording of multithreaded processors lecture](#)
 - [Audio recording of trace processors lecture](#)
 - [Evaluation of Design Options for the Trace Cache Fetch Mechanism](#)
 - [Exploiting Choice: Instruction Fetch and Issue on an Implementable Simultaneous Multithreading Processor](#)
 - [Exploiting Instruction Level Parallelism in Processors by Caching Scheduled Groups](#)
 - [Multiscalar processors](#)
 - [Multithreaded Processors](#)
 - [Multithreaded processors](#)
 - [Simultaneous multithreading: Maximizing on-chip parallelism](#)
 - [Single-ISA Heterogeneous Multi-Core Architectures for Multithreaded Workload Performance](#)
 - [Single-Program Speculative Multithreading \(SPSM\) Architecture: Compiler-assisted Fine-Grained Multithreading](#)
 - [The Microarchitecture of the Pentium 4 Processor](#)
 - [The Vector-Thread Architecture](#)
 - [The block-based trace cache](#)
 - [Trace Caches and Trace Processors](#)
 - [Trace cache: a low latency approach to high bandwidth instruction fetching](#)
 - [Trace processors](#)
-

Year 2004 Technical Training

Multithreaded processors

Author: Jamie H. Moreno

[MT_procs.pdf](#)

April 29, 2004 [32 Pages]

Slides from multithreaded processors lecture.

Audio recording of multithreaded processors lecture

Multithreading_lecture_audio_2hrs_42min.mp3

April 29, 2004

Trace Caches and Trace Processors

Author: Ravi nair

TraceProcs.pdf

April 29, 2004 [38 Pages]

Slides from trace processors lecture.

Audio recording of trace processors lecture

Trace_Processors_lecture_audio_2hrs_25min.mp3

April 29, 2004

References from Trace processors lecture

Trace cache: a low latency approach to high bandwidth instruction fetching

Author: Rotenberg, E., Bennett, S., and Smith, J.E.

00566447.pdf

[11 Pages]

As the issue width of superscalar processors is increased, instruction fetch bandwidth requirements will also increase. It will become necessary to fetch multiple basic blocks per cycle. Conventional instruction caches hinder this effort because long instruction sequences are not always in contiguous cache locations. We propose supplementing the conventional instruction cache with a trace cache. This structure caches traces of the dynamic instruction stream, so instructions that are otherwise noncontiguous appear contiguous. For the Instruction Benchmark Suite (IBS) and SPEC92 integer benchmarks, a 4 kilobyte trace cache improves performance on average by 28% over conventional sequential fetching. Further it is shown that the trace cache's efficient, low latency approach enables it to outperform more complex mechanisms that work solely out of the instruction cache.

Trace processors

Author: Rotenberg, E., Jacobson, Q., Sazeides, Y., and Smith, J.
[00645805.pdf](#)
[11 Pages]

Traces are dynamic instruction sequences constructed and cached by hardware. A microarchitecture organized around traces is presented as a means for efficiently executing many instructions per cycle. Trace processors exploit both control flow and data flow hierarchy to overcome complexity and architectural limitations of conventional superscalar processors by (1) distributing execution resources based on trace boundaries and (2) applying control and data prediction at the trace level rather than individual branches or instructions. Three sets of experiments using the SPECInt95 benchmarks are presented. (i) A detailed evaluation of trace processor configurations: the results affirm that significant instruction-level parallelism can be exploited in integer programs (2 to 6 instructions per cycle). We also isolate the impact of distributed resources, and quantify the value of successively doubling the number of distributed elements. (ii) A trace processor with data prediction applied to inter-trace dependences: potential performance improvement with perfect prediction is around 45% for all benchmarks. With realistic prediction, gcc achieves an actual improvement of 10%. (iii) Evaluation of aggressive control flow: some benchmarks benefit from control independence by as much as 10%.

The block-based trace cache

Author: Black, B., Rychlik, B., and Shen, J.P.
[00765951.pdf](#)
[12 Pages]

The trace cache is a recently proposed solution to achieving high instruction fetch bandwidth by buffering and reusing dynamic instruction traces. This work presents a new block-based trace cache implementation that can achieve higher IPC performance with more efficient storage of traces. Instead of explicitly storing instructions of a trace, pointers to blocks constituting a trace are stored in a much smaller trace table. The block-based trace cache renames fetch addresses at the basic block level and stores aligned blocks in a block cache. Traces are constructed by accessing the replicated block cache using block pointers from the trace table. Performance potential of the blockbased trace cache is quantified and compared with perfect branch prediction and perfect fetch schemes. Comparing to the conventional trace cache, the block-based design can achieve higher IPC, with less impact on cycle time.

Exploiting Instruction Level Parallelism in Processors by Caching Scheduled Groups

Author: Ravi Nair and Martin E. Hopkins
[00604516.pdf](#)
[13 Pages]

The Microarchitecture of the Pentium 4 Processor

Author: Glenn Hinton, Dave Sager, Mike Upton, Darrell Boggs, Doug Carmean, Alan Kyker, and Patrice Roussel

[Hinton_etal_-_Intel_TJ_2001_-_Pentium_4.pdf](#)
[Pages]

Evaluation of Design Options for the Trace Cache Fetch Mechanism

Author: Sanjay Jeram Patel, Daniel Holmes Friendly, and Yale N. Patt
[00752661.pdf](#)
[12 Pages]

In this paper, we examine some critical design features of a trace cache fetch engine for a 16-wide issue processor and evaluate their effects on performance. We evaluate path associativity, partial matching, and inactive issue, all of which are straightforward extensions to the trace cache. We examine features such as the fill unit and branch predictor design. In our final analysis, we show that the trace cache mechanism attains a 28 percent performance improvement over an aggressive single block fetch mechanism and a 15 percent improvement over a sequential multiblock mechanism.

References from multithreaded processor lecture

Multiscalar processors

Author: Sohi, G.S., Breach, S.E., Vijaykumar, T.N.
[00524580.pdf](#)
22-24 June 1995 [12 Pages]

Multiscalar processors use a new, aggressive implementation paradigm for extracting large quantities of instruction level parallelism from ordinary high level language programs. A single program is divided into a collection of tasks by a combination of software and hardware. The tasks are distributed to a number of parallel processing units which reside within a processor complex. Each of these units fetches and executes instructions belonging to its assigned task. The appearance of a single logical register file is maintained with a copy in each parallel processing unit. Register results are dynamically routed among the many parallel processing units with the help of compiler-generated masks. Memory accesses may occur speculatively without knowledge of preceding loads or stores. Addresses are disambiguated dynamically, many in parallel, and processing waits only for true data dependences. This paper presents the philosophy of the multiscalar paradigm, the structure of multiscalar programs, and the hardware architecture of a multiscalar processor. The paper also discusses performance issues in the multiscalar model, and compares the multiscalar paradigm with other paradigms. Experimental results evaluating the performance of a sample of multiscalar organizations are also presented.

Simultaneous multithreading: Maximizing on-chip parallelism

Author: Tullsen, D.M., Eggers, S.J., and Levy, H.M.
[00524578.pdf](#)
22-24 June 1995 [12 Pages]

This paper examines simultaneous multithreading, a technique permitting several independent threads to issue instructions to a superscalar's multiple functional units in a single cycle. We present several models of simultaneous multithreading and compare them with alternative organizations: a wide superscalar, a fine-grain multithreaded processor, and single-chip, multiple-issue multiprocessor architectures. Our results show that both (single-threaded) superscalar and fine-grain multithreaded architectures are limited in their ability to utilize the resources of a wide-issue processor. Simultaneous multithreading has the potential to achieve 4 times the throughput of a superscalar, and double that of fine-grain multi-threading. We evaluate several cache configurations made possible by this type of organization and evaluate tradeoffs between them. We also show that simultaneous multithreading is an attractive alternative to single-chip multiprocessors; simultaneous multithreaded processors with a variety of organizations outperform corresponding conventional multiprocessors with similar execution resources. While simultaneous multithreading has excellent potential to increase processor utilization, it can add substantial complexity to the design. We examine many of these complexities and evaluate alternative organizations in the design space.

Note: Section 6 of this document discusses advantages of SMT over multi-processors.

Single-Program Speculative Multithreading (SPSM) Architecture: Compiler-assisted Fine-Grained Multithreading

Author: Pradeep K. Dubey, Kevin O'Brien, Kathryn M. O'Brien, and Charles Barton

[Dubey_SPSM.pdf](#)

Feb. 6, 1995 [32 Pages]

Single-ISA Heterogeneous Multi-Core Architectures for Multithreaded Workload Performance

Author: Rakesh Kumar, Dean M. Tullsen, Parthasarathy Ranganathan, Norman P. Jouppi, Keith I. Farkas

[Kumar_ISCA04.pdf](#)

June 2004 [12 Pages]

Exploiting Choice: Instruction Fetch and Issue on an Implementable Simultaneous Multithreading Processor

Author: Dean M. Tullsen, Susan J. Eggers, Joel S. Emer, and Henry M. Levy

[Tullsen_ISCA96.pdf](#)

May 1996 [12 Pages]

Multithreaded Processors

Author: Teho Ungerer, Borut Robic, nd Juru Silc

[Ungerer_Computer_Journal.pdf](#)

2002 [29 Pages]

A Survey of Processors with Explicit Multithreading

Author: Theo Ungerer, Borut Robic, and Jurij Silc
[Ungerer Computing Surveys.pdf](#)
March 2003 [35 Pages]

The Vector-Thread Architecture

Author: Ronny Krashinsky, Christopher Batten, Mark Hampton, Steve Gerding, Brian Pharris, Jared Casper, and Krste Asanovic
[Vector_Thread_ISCA04.pdf](#)
June 2004 [12 Pages]

A multithreaded PowerPC processor for commercial servers

Author: J. M. Borkenhagen, R. J. Eickemeyer, R. N. Kalla, and S. R. Kunkel
[borkenhagen_RS64_IV.pdf](#)
November 2000 [14 Pages]

